Computational Materials Science 125 (2016) 123–135

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Critical assessment of regression-based machine learning methods for polymer dielectrics

Arun Mannodi-Kanakkithodi [a], Ghanshyam Pilania [b], Rampi Ramprasad [a,*]

[a] Department of Materials Science and Engineering, Institute of Materials Science, University of Connecticut, 97 North Eagleville Road, Storrs, CT 06269, USA
[b] Materials Science and Technology Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

ARTICLE INFO

Article history:
Received 6 June 2016
Received in revised form 19 August 2016
Accepted 25 August 2016

Keywords:
Materials informatics
Density functional theory
Regression

ABSTRACT

The design of new and improved materials for different applications of interest is boosted by combining computations or experiments with machine learning techniques. Materials scientists seek to use learning algorithms that can easily and efficiently be applied to their data in order to obtain quantitative property prediction models. Here, we utilize a first principles generated dataset of the electronic and dielectric properties of a chemical space of polymers to test different kinds of regression algorithms used by the machine learning community today. We explore several possibilities for the hyper-parameters that go into such learning schemes, and establish optimal strategies and parameters for high-fidelity polymer dielectrics property prediction models.

Published by Elsevier B.V.

1. Introduction

Materials science has greatly benefited in recent times from the application of machine learning techniques to available or newly generated materials data [1–7]. Whereas accurate computations and careful experimental measurements are the standard treatments for new materials discovery, machine learning can accelerate the process significantly, and open up opportunities for exploring complex chemical spaces efficiently. Given any robust dataset of materials properties, learning approaches involve making correlations and mappings between crucial but easily accessible features of materials on the one hand, and the properties of interest on the other. Relationships formed between features and properties can then be exploited for making qualitative, semi-quantitative or quantitative predictions on unseen materials.

In a recent study of designing dielectric polymers for energy storage applications [1,8], we applied machine learning techniques on computational data to develop property prediction models. With respect to high energy density capactitors, polymers suitable to be used as dielectrics should show a high dielectric constant and a large band gap, amongst other crucial features [9,10]. We used density functional theory (DFT) computations to generate dielectric constant (divided into two components, the electronic and the ionic contributions) and band gap data for a selected chemical space of organic polymers. These polymers were built by simple linear combinations of chemical units opted out of a pool of 7 basic blocks: CH_2, NH, CO, C_6H_4, C_4H_2S, CS, and O. For each polymer, DFT helps determine the ground state crystalline arrangement, for which the properties are then computed using known formalisms. Validation of the computed properties against experimental measurements [9] for known polymers makes this a reliable methodology.

Once the property data was generated for 284 polymers (all this data is presented in Refs. [1,8]), it was possible to perform machine learning via an intermediate polymer fingerprinting step. The fingerprint is mapped to the properties—the band gap (in eV), the electronic dielectric constant, and the ionic dielectric constant—to develop an efficient prediction model, that will give as output the properties of any new polymer by converting it into its fingerprint. Once a reasonably accurate prediction model is trained, one can instantly predict the dielectric constants and band gaps of any new polymers that were not considered during computations, thus providing an accelerated materials design route.

Apart from the availability of robust, uniformly generated data, there are a number of other essential factors in the machine learning process that need to be taken care of for optimal learning. These include defining a suitable fingerprint, choosing a learning algorithm, and determining the necessary subset of the data that is needed for training the learning model [5]. The fingerprints we chose and tested in Ref. [1] were chemo-structural in nature, that is, they quantified the types and combinations of different constituent blocks in the polymer. Three fingerprints were used: a count of the different types of building blocks in the polymer,

* Corresponding author.
E-mail address: rampi.ramprasad@uconn.edu (R. Ramprasad).

http://dx.doi.org/10.1016/j.commatsci.2016.08.039
0927-0256/Published by Elsevier B.V.

called fingerprint M_I, a count of the types of block pairs (fingerprint M_{II}), and a count of the types of block triplets (fingerprint M_{III}). The fingerprints were normalized and generalized for any number of blocks in the polymer repeat unit, and used to train a regression model for the three properties of interest.

Whereas all three fingerprints were tested in Ref. [1], the learning algorithm used was Kernel Ridge Regression (KRR) [11]—a nonlinear regression technique that works on the principle of similarity. Euclidean distances between fingerprints were used to quantify the similarity. A distance kernel goes into the definition of the property here, for which a Gaussian kernel was used. Around 90% of the entire polymer dataset was used to train the KRR model, and predictions were made on the remaining points as a test of the performances. Mean absolute errors (MAE) in prediction of less than 10% with respect to the DFT values were seen, which is satisfactory for a statistical model and the best performance that could be obtained using the current optimal learning parameters. The optimal fingerprint used here was M_{III}, with M_{II} and M_I discarded owing to larger prediction errors.

Although we obtained learning models as described above to predict polymer properties with reasonable accuracies, a detailed study of all the different possible machine learning (or regression) parameters is due. Such a study can be very valuable in terms of truly testing the capabilities of our machine learning philosophy for the given polymer dataset, and indeed, improving the performances. In Table 1, we try to capture all these different parameters, mentioning the specific choices that we used in Ref. [1] as well as the other possible options explored here. Whereas the fingerprint choices were already rigorously tested, each of the other parameters provide room for further testing, and thus possible performance improvement.

In this paper, we take the same polymer dataset and analyze the machine learning prediction performances for different regression algorithms, different distance kernel choices, different training set sizes and different error definitions. Possible alternative algorithms to KRR include, but are not limited to: Linear Regression (LR), Support Vector Regression (SVR), Gaussian Process Regression (GPR) and SVR with AdaBoost. Whereas we used KRR with a Gaussian kernel in Ref. [1], Linear, Laplacian or Polynomial kernels can be used as alternatives in any kernel-based regression algorithm. Further, the training set size can be varied systematically to study the prediction errors. The prediction errors can be quantified in different ways, such as mean absolute error (MAE), root mean square error (RMSE) and error based on the coefficient of determination ($1 - R^2$).

In the following sections, we present our results and discussions based on the analysis of all these machine learning parameters. We attempt to compare them critically with each other, and comment on the best possible combination of parameters that must be used given the present polymer dataset.

Table 1
A comparison of various choices of machine learning parameters used in Ref. [1] and explored here. The acronyms used stand for: Kernel Ridge Regression (KRR), Support Vector Regression (SVR), Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and goodness of fit (R^2).

Machine learning parameters	Choices used in Ref. [1]	Choices explored here
Fingerprint	M_I, M_{II}, M_{III}	M_{III}
Regression algorithm	KRR	KRR, SVR, AdaBoost
Type of kernel	Gaussian	Gaussian, Laplacian, linear, polynomial
Training set size	90% of Data	Learning curves
Error definition	MAE	RMSE, $1 - R^2$

2. Kernel Ridge Regression (KRR)

In this section, we delve deeper into KRR, the algorithm that formed the basis of all machine learning prediction models in Ref. [1]. KRR is a similarity based regression algorithm that inherently takes the nonlinearity of the system into account. The 'similarity' between any two data points is defined using some standard mathematical measure of distance, such as a Euclidean distance. For any two polymers i and j having fingerprints $\vec{\mathbf{x}}_i$ and $\vec{\mathbf{x}}_j$ respectively (where $\vec{\mathbf{x}}_i$ is an m dimensional vector with components $x_i^1, x_i^2, x_i^3 \ldots x_i^m$), the Euclidean distance between them will be defined as:

$$d(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \|\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j\|_2 = \sqrt{\left(x_i^1 - x_j^1\right)^2 + \left(x_i^2 - x_j^2\right)^2 + \ldots + \left(x_i^m - x_j^m\right)^2}. \quad (1)$$

The smaller (larger) is this distance, the more similar (dissimilar) the two polymers are. Now, KRR involves defining the property of interest (the output) as a function of such a distance measure, so that the property of any polymer can be estimated by taking its distances from all the other polymers. Mathematically, the predicted property of polymer j, denoted by $P(j)$, will be defined as follows:

$$P_{pred}(j) = \sum_{i=1}^{n} \alpha_i \mathcal{K}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j). \quad (2)$$

The summation is performed over the entire training set size n, and $\mathcal{K}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j)$ is the *kernel function* that is defined in terms of $d(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j)$, the distance between polymer i (in the training set) and polymer j. The purpose of the kernel function is to transform the points (the polymers) from the fingerprint space to a higher dimensional space, thus making nonlinear mapping possible [12]. The two crucial parameters that need to be optimized here are the kernel coefficients α_i and the parameters that go into the kernel definition—such as the Gaussian width for a Gaussian kernel. Training of a KRR model essentially involves an iterative minimization of prediction errors leading to the optimal parameter choices.

In practice, as mentioned in the Introduction, the total available dataset is divided into two parts—the training dataset and the test dataset. When training the model using the former, an important step that must be carried out is *cross-validation*, wherein the training set itself is divided into a number of subsets. One of the subsets is used as a temporary test set while training is performed on the remaining subsets, and this procedure is repeated for each of the subsets. The optimal regression parameters are obtained corresponding to minimum average prediction errors on the temporary test sets; subsequently, the error computed over the entire training set with these parameters is referred to as the 'cross-validation error', or sometimes the cross-validated 'training error'. The purpose of cross-validation is to avoid overfitting in the data and to make the model more generalizable—that is, to ensure that the model predictions would work reasonably for points outside the training dataset.

Mathematically, the training process involves a minimization of the following expression:

$$\underset{\alpha_1,\ldots,\alpha_n}{\arg\min} \sum_{i=1}^{n} (P_{pred}(i) - P_{actual}(i))^2 + \lambda \sum_{i=1}^{n} \|\alpha_i\|_2^2. \quad (3)$$

where $P_{pred}(i)$ is the KRR model predicted property value of polymer i as defined in Eq. (2) and $P_{actual}(i)$ is its actual property value; $(P_{pred}(i) - P_{actual}(i))$ is thus a measure of the prediction error. However, the second term in the expression involves the regularization parameter λ. Regularization [2] is an important step that is again aimed at preventing overfitting, and involves adding extra

information to the expression being minimized. The solution to Eq. (3) is given by

$$\vec{\boldsymbol{\alpha}} = (\mathbf{K}_{train} + \lambda \mathbf{I})^{-1} \cdot \vec{\mathbf{P}}_{actual}. \quad (4)$$

where $\vec{\boldsymbol{\alpha}}$ is the vector of all α_i values, $\mathbf{K}_{train}$ represents the kernel matrix for the entire training set, and P_{actual} represents the vector of actual property values for all points in the training set. Based on the above discussion, it would appear that the two important parameters that need to be optimized during the training process are the following: regularization parameter λ, and the relevant kernel parameters. A set of values for these parameters are tested here towards the minimization of the expression in Eq. (3), thus yielding the final form of Eq. (2) that can be used for predictions on the test set.

2.1. Learning with different kernels

When applied in Eq. (3), any kernel function will be expressed in terms of the distance between two polymers as defined by Eq. (1). Whereas a given polymer i exists as x_i in the fingerprint space, implementing a kernel function is simply a way of projecting the polymer to the kernel space, which is what makes the application of a technique such as kernel ridge regression possible. Many different types of kernel definitions can be applied in Eq. (2), as shown in Table 1, such as a linear kernel, polynomial kernel, Gaussian kernel, and Laplacian kernel. Here, we consider three different kinds of kernels and compare the KRR prediction performances with each, with prediction errors given using two error definitions: the root mean square error (RMSE) and $1 - R^2$, where R^2 is known as the coefficient of determination and represents goodness of the fit. We should thus be able to determine the best performing kernel with respect to one regression algorithm: KRR.

2.1.1. Gaussian kernels

A Gaussian kernel (an example of a radial basis function kernel) is defined for any two polymers i and j as:

$$\mathcal{K}_{\mathcal{G}}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = exp\left(\frac{-\|\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j\|_2^2}{2\sigma^2}\right). \quad (5)$$

Here, the numerator inside the exponential term contains the Euclidean distance measure, or the L_2-norm, and the denominator contains σ, a kernel parameter known as the Gaussian width. One of the most important things to note here is that σ is an adjustable parameter that affects the kernel performance in a major way. Given the square scaling relationship, even a slight overestimation of σ can cause the exponential to start acting linearly, which leads to a loss in nonlinearity of the kernel projection and thus, the KRR algorithm. On the other hand, an underestimation of σ can lead to overfitting in the training data and consequently, poor prediction performances on the test set. Estimating the optimal σ value is thus of utmost importance, and the two parameters that need to be optimized while performing KRR with a Gaussian kernel are λ and σ.

2.1.2. Laplacian kernels

A Laplacian kernel is also a radial basis function kernel, and can be expressed mathematically as:

$$\mathcal{K}_{\mathcal{L}}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = exp\left(\frac{-\|\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j\|_1}{\sigma}\right). \quad (6)$$

The distance measure in the numerator of the exponential term here is the Manhattan distance, or the L_1-norm. The observations made about σ in the discussion of Gaussian kernels are applicable here as well.

2.1.3. Polynomial kernels

Whereas the two kernels described above are exponential functions, yet another choice for a kernel could be a polynomial function. Such a kernel can be expressed as follows

$$\mathcal{K}_{poly}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = (\gamma\langle\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j\rangle + c)^d. \quad (7)$$

where $\langle\cdot,\cdot\rangle$ denotes the dot product in the space of the input feature vectors and the adjustable parameters are the constant term c and the degree of the polynomial, d. Here, we take $c = 0$ and $\gamma = 1$ for simplicity, which leaves d as the one important kernel parameter to be optimized.

Figs. 1–3 show plots between two vital KRR parameters when using the Gaussian, Laplacian and polynomial kernel respectively. Whereas the plot is between σ and λ in Figs. 1 and 2 (on a logarithmic scale), the plot in Fig. 3 is between parameter d and λ. Shown in different colors (according to the adjoining color bar) in each of the plots, for the three properties and using two different error definitions, are the respective prediction errors corresponding to any combination of the two parameters. The prediction errors are estimated for the training set points and the test set points respectively, as the averaged RMSE or $(1 - R^2)$ errors over all the points, and the test errors are depicted in Figs. 1–3.

The plots in Figs. 1–3 enable us to determine regions of unfavorable parameter values as well as the region where the optima will be found. For example, a combination of $\sigma = 4$ and $\lambda = 2^{-7}$ appears to provide the minimum $1 - R^2$ and RMSE errors for band gap predictions. The optimal $[\sigma, \lambda]$ or the optimal $[d, \lambda]$ values can similarly be obtained for KRR models for each property, using each kind of kernel. The lowest training and test prediction errors thus observed for the optimal parameter choices with the three different kernels are listed in Table 2. It should be noted that the optimal λ values obtained using the polynomial kernel, especially for the ionic dielectric constant and the band gap, are many orders of magnitude smaller than those with the two exponential kernels. This has important consequences, as we explain below.

The plots of most interest following this study are the ones presented in Fig. 4. KRR performances using the three kernels (based on the optimal parameter choices for each) are shown here for the three properties—electronic dielectric constant, ionic dielectric constant or band gap—in the form of parity plots between the KRR predicted values and the actual DFT values. Fig. 4a shows the KRR performances using a Gaussian kernel, which is the same as the machine learning models that were presented in Ref. [1]. It can be seen that there is no clear improvement in the prediction performances on the test set points upon going from the Gaussian to the Laplacian (Fig. 4b) and the polynomial (Fig. 4c) kernels, which vindicates the prior usage of the Gaussian kernel.

For the electronic dielectric constant, the performance worsens with the polynomial kernel when compared to the exponential kernels. For the two other properties, whereas the test prediction errors are more or less the same with every kernel, there is a problem of overfitting in the data to some extent with the Laplacian kernel, but to a large extent with the polynomial kernel. This is because of the smaller values of the regularization parameter λ as pointed out earlier, which leads to a shrinkage of the second term in Eq. (3). Given that our model selection is based on the lowest cross-validation errors that can be obtained from the training set, the scatter in the test set points seen in Fig. 4c shows the inadequacy of the polynomial kernel in representing the properties as a function of the fingerprint. This further points towards the exponential kernels, and specifically the Gaussian kernel, being the best choice for KRR among the kernels and applications considered here.

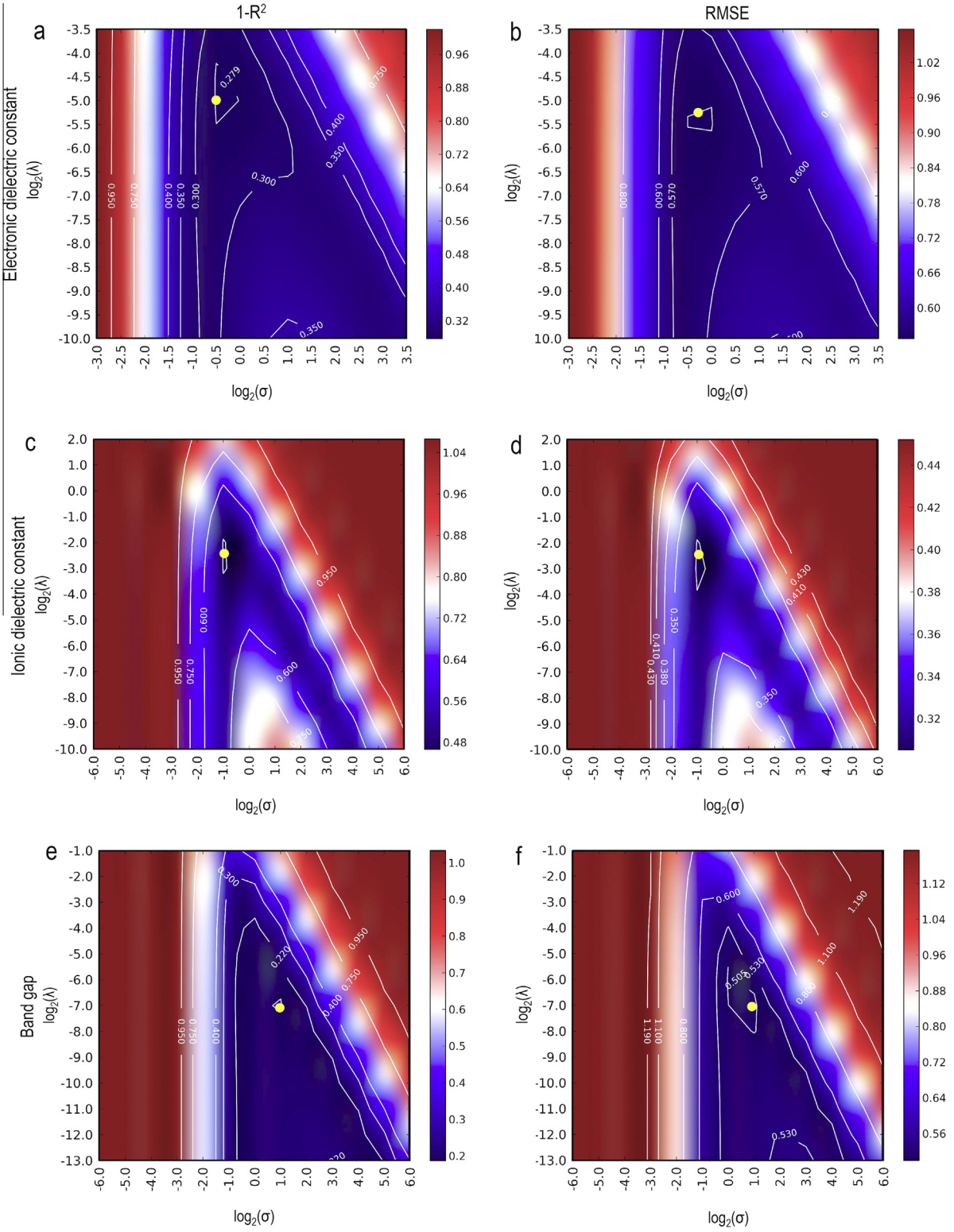

Fig. 1. Optimal parameter selection for the KRR ML models with Gaussian kernels.

2.2. Optimal training set size: learning curves

While we have considered a training set size of 250 (approximately 90% of the entire dataset) in all the analyses presented so far, a rigorous demonstration of how we obtained this particular figure—or if 250 is indeed the optimal training set size—is missing. In any statistical learning treatment, determining the minimum number of data points necessary for training a satisfactory model is of utmost importance. One may not possess sufficient data to train a respectable model, or one may possess excess data, in which case some points can safely be put aside for model testing purposes. Here, we present a systematic study of the adequacy of

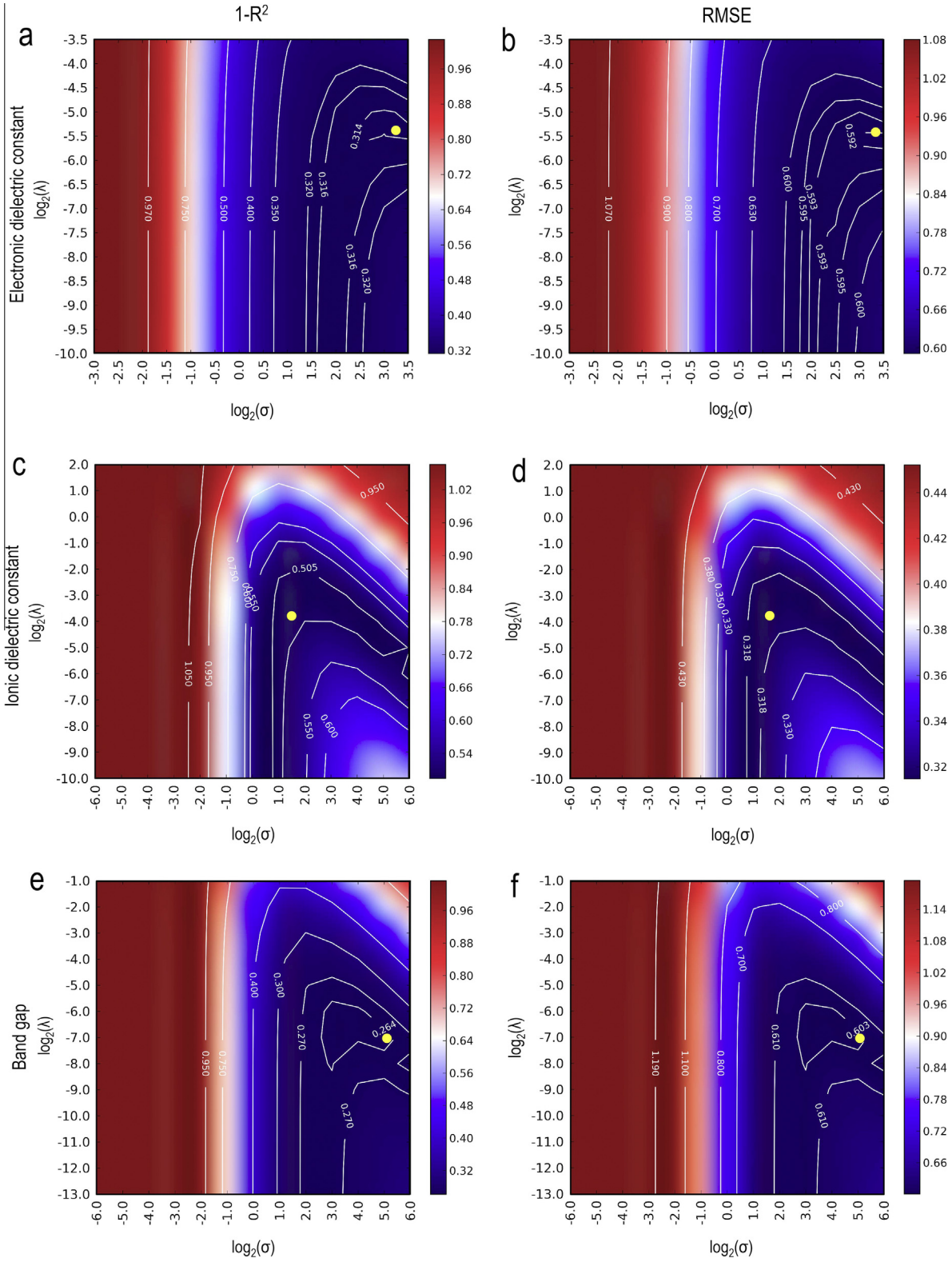

Fig. 2. Optimal parameter selection for the KRR ML models with Laplacian kernels.

the training data set with respect to obtaining acceptable statistical prediction errors, using KRR with the three different kinds of kernels as before.

Shown in Fig. 5 for the three properties, for KRR with each kernel, are plots between the prediction errors $(1 - R^2)$ and the training set sizes, referred to in machine learning practices as *learning curves*. [7] We increase the training set size from 50 ($\sim 20\%$ of the dataset of 284) in steps of 5% of the entire dataset, all the way to 250 ($\sim 90\%$ of the dataset); the test set is, of course, all the remaining points in the dataset. In each of the 9 cases, we consider 50 different randomly chosen training set populations for a given training set size, and measure the prediction errors (errors

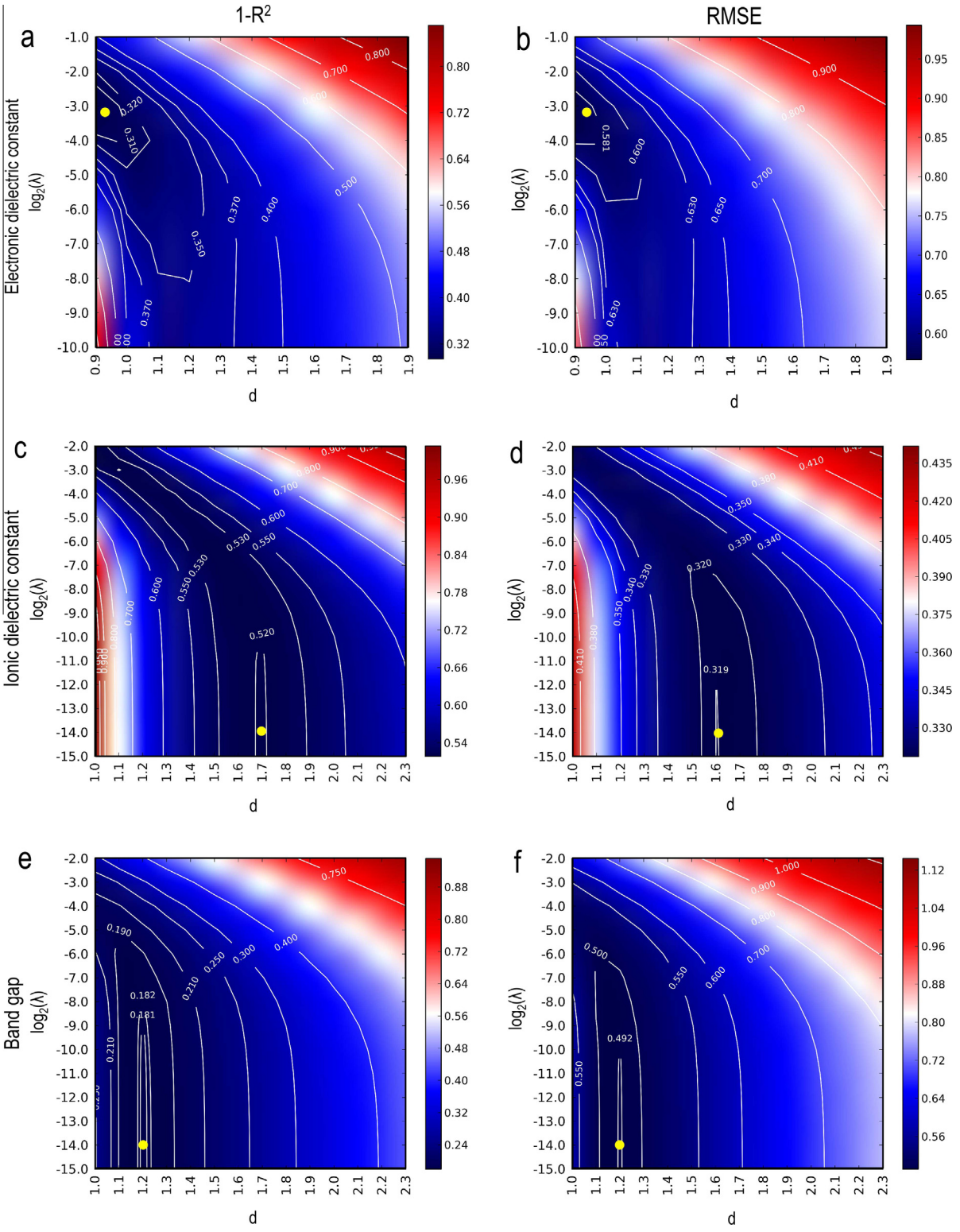

Fig. 3. Optimal parameter selection for the KRR ML models with polynomial kernels.

on the test set points) using the respective trained models. What we have plotted in Fig. 5 are the averaged test set prediction errors as well as the standard deviation in errors, for different training sizes.

As one would expect, the general trend exhibited in each of the plots is a gradual decrease in the average error as the training set size increases, which is simply owing to the improvement of the prediction model with a higher number of points trained upon. Whereas the standard deviations do not necessarily decrease the same way, the maximum and minimum errors that are seen generally follow the same trend as the average errors. In fact, the standard deviations seem to be higher in many of the cases for a large training set size, which happens because while some prediction models are excellent (reflected in the low minimum prediction

Table 2
Training and test prediction errors (RMSE and $1-R^2$) with all the regression algorithms. ϵ represents dielectric constant. The reported *rms* error for band gaps is in eV.

Learning algorithm	Kernel used	*rms* error			$1-R^2$		
		ϵ-electronic	ϵ-ionic	Bandgap	ϵ-electronic	ϵ-ionic	Bandgap
Performance on test set							
KRR	Gaussian	0.366	0.162	0.506	0.193	0.368	0.203
KRR	Laplacian	0.457	0.161	0.519	0.187	0.361	0.214
KRR	Polynomial	0.510	0.171	0.494	0.233	0.407	0.194
SVR	Gaussian	0.527	0.136	0.480	0.250	0.259	0.183
SVR + Boosting	Gaussian	0.465	0.325	0.494	0.310	0.667	0.155
Performance on training set							
KRR	Gaussian	0.112	0.107	0.196	0.013	0.051	0.026
KRR	Laplacian	0.118	0.038	0.141	0.016	0.006	0.013
KRR	Polynomial	0.292	0.000	0.001	0.097	0.000	0.000
SVR	Gaussian	0.292	0.195	0.097	0.098	0.171	0.006
SVR + Boosting	Gaussian	0.256	0.185	0.062	0.070	0.160	0.003

errors observed), there could be possible data overfitting in some others given the few remaining points that constitute the test set may not be well predictable (reflected in the high maximum prediction errors observed). The average errors steadily decrease all the way to a training set size of 250, which justifies our optimal training size selection in Ref. [1], with the exception of band gap predictions with Gaussian kernel where the error minimum occurs around 220 points.

The learning curves are, for most parts, very smooth in nature and follow the average decreasing trend we expect. The standard deviations we observe are a consequence of the dataset at hand, where selection of the appropriate 'number' as well as 'nature' of training set points has a strong effect on the prediction model. For instance, the absence of certain combinations of constituent polymer chemical blocks (as reflected in the polymer fingerprints) in the training set would make predictions on the test set containing such polymers quite poor, despite perhaps a large number of data points being present in the training set. This is what leads to a high standard deviation in prediction errors in some cases; nevertheless, the learning curves do tell us that a large enough training set size would enable us to train regression models with sufficiently low prediction errors.

3. Support Vector Regression (SVR)

While Kernel Ridge Regression has provided reasonable prediction accuracies so far, the machine learning community has been known to use many other learning algorithms with varying degrees of success. One such algorithm is Support Vector Machines (SVM), supervised learning techniques developed at AT&T Bell Laboratories by Vapnik and co-workers [13,14] and widely used in classification problems. When applied to regression and function estimation problems, SVMs are called Support Vector Regression (SVR) and constitute a very popular regression algorithm which is implemented in most of the standard machine learning packages. SVMs are efficient tools for going beyond linear classification or regression owing to the implementation of the 'kernel trick', which as explained earlier, simply involves transforming data points to a higher dimensional kernel induced feature space to incorporate nonlinearity [12].

Given the input variables (the polymer fingerprint) and the response variable (the polymer property), in the form of training data $\{(\vec{\mathbf{x}}_1, y_1), \ldots, (\vec{\mathbf{x}}_n, y_n)\} \subset \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ denotes a d-dimensional feature space (*e.g.*, $\mathcal{X} = \mathbb{R}^d$), an ϵ-SVR algorithm tries to find a function $f(x)$ that has at most ϵ deviation form the targeted property values y_i, and at the same time is as flat as possible. Any deviation larger than ϵ is not acceptable.

For linear regression, the function $f(x)$ can take the following form:

$$f(x) = \langle w, x \rangle + b, \tag{8}$$

where $w \in \mathcal{X}, b \in \mathbb{R}$. *Flatness* of the function $f(x)$ in this case means that we seek a vector w with a small norm, *i.e.* $\|w\|^2 = \langle w, w \rangle$. The convex optimization problem can then be written as:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 \\ &\text{subject to} && y_i - \langle w, \vec{\mathbf{x}}_i \rangle - b \leqslant \epsilon \\ & && \langle w, \vec{\mathbf{x}}_i \rangle + b - y_i \leqslant \epsilon. \end{aligned} \tag{9}$$

In writing the above expression, we tacitly assume that the convex optimization problem is feasible, or in other words, there exists a function f that approximates all training pairs $(\vec{\mathbf{x}}_i, y_i)$ with at least ϵ precision. However, in practice, this may not be the case many a times, and we have to allow for some errors. This is done by incorporating a "soft margin" loss function through slack variables ξ_i and ξ_i^* in the otherwise infeasible optimization problem. Introduction of the slack variables in Eq. (9) leads to the following formulation:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ &\text{subject to} && y_i - \langle w, \vec{\mathbf{x}}_i \rangle - b \leqslant \epsilon + \xi_i \\ & && \langle w, \vec{\mathbf{x}}_i \rangle + b - y_i \leqslant \epsilon + \xi_i^* \\ & && \xi_i, \xi_i^* \geqslant 0. \end{aligned} \tag{10}$$

where the positive constant C determines the trade-off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated through the slack variables. The objective presented in the above minimization problem (Eq. (10)) is also referred to as the *primal* objective function, which is solved by constructing a Lagrange function $\mathcal{L}$, and accounting for the constraints through the positively constrained Lagrange multipliers $\alpha_i, \alpha_i^*, \beta_i$ and β_i^*, as follows:

$$\begin{aligned} \mathcal{L}: \\ &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) - \sum_{i=1}^{n}\alpha_i\left(\epsilon + \xi_i - y_i + \langle w, \vec{\mathbf{x}}_i \rangle + b\right) \\ &\quad - \sum_{i=1}^{n}\alpha_i^*\left(\epsilon + \xi_i^* + y_i - \langle w, \vec{\mathbf{x}}_i \rangle - b\right) - \sum_{i=1}^{n}(\beta_i\xi_i + \beta_i^*\xi_i^*). \end{aligned} \tag{11}$$

In accordance with the saddle point condition, the partial derivatives of $\mathcal{L}$ with respect to the variables w, b, ξ_i, and ξ_i^* lead to linear equations which when substituted back into Eq. (11) lead to the so called dual optimization problem of SVR, as given below,

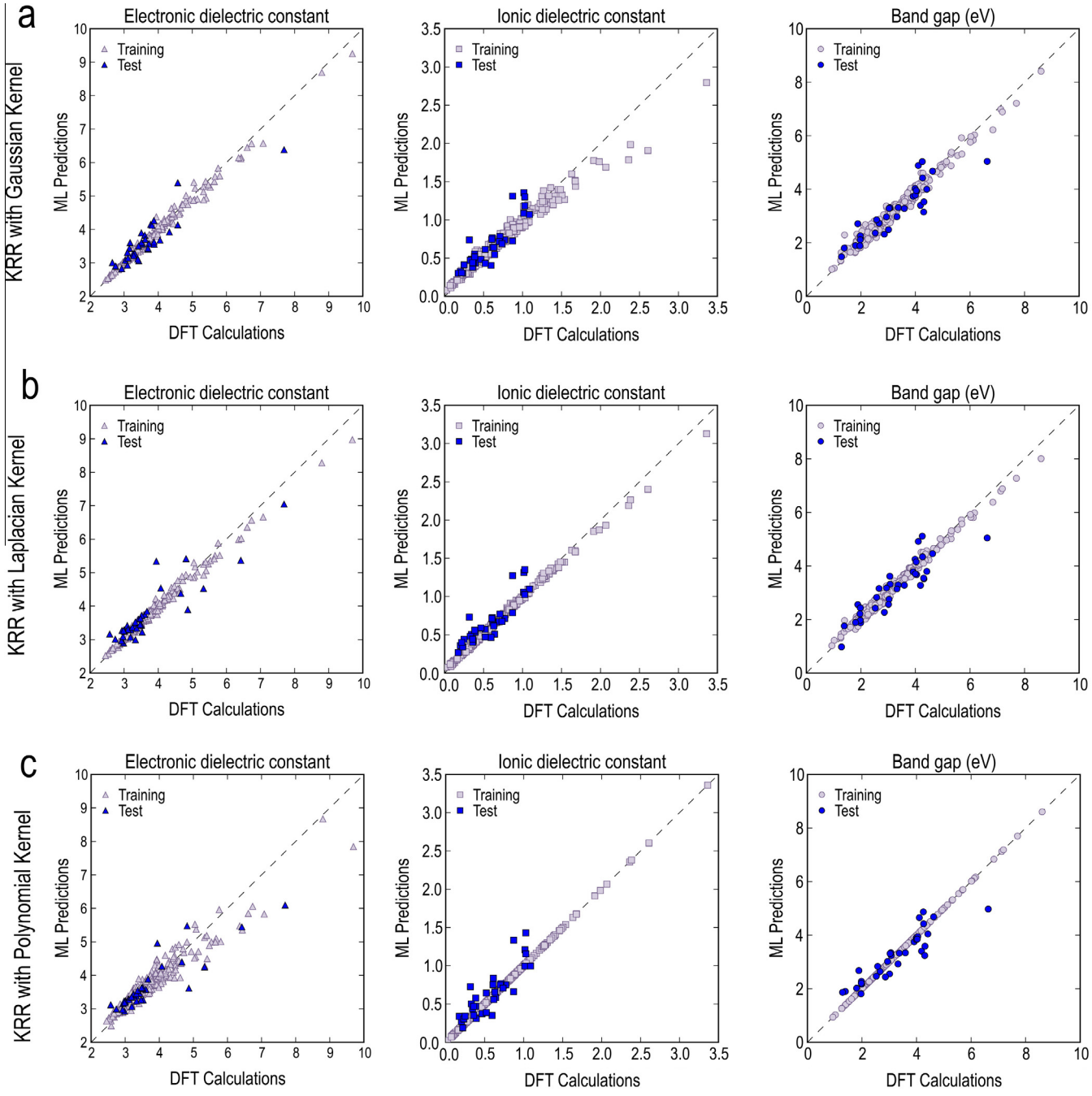

Fig. 4. Learning and prediction performances of the KRR ML models with different kernels.

$$\begin{aligned} \text{maximize} \quad & -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)\left(\alpha_j-\alpha_j^*\right)\langle\vec{\mathbf{x}}_i,\vec{\mathbf{x}}_j\rangle \\ & -\epsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{n}y_i(\alpha_i-\alpha_i^*) \qquad (12) \\ \text{subject to} \quad & \sum_{i,j=1}^{n}(\alpha_i-\alpha_i^*)=0 \quad \text{and} \quad \alpha_i,\alpha_i^*\in[0,C]. \end{aligned}$$

By solving the above dual optimization problem, α_i, α_i^* and b can be determined, which can then be used to make predictions on new systems with a given input x as:

$$f(x)=\sum_{i=1}^{n}(\alpha_i-\alpha_i^*)\langle\vec{\mathbf{x}}_i,x\rangle+b. \qquad (13)$$

Note that once the α_is and α_i^*s have been determined, w can be written as $\sum_{i=1}^{n}(\alpha_i-\alpha_i^*)$. This is known as *Support Vector expansion*, which describes w as a linear combination of the training data points. At this point it is important to note that the only data points for which the Lagrange multipliers (*i.e.*, either α_i or α_i^*; both of them can not be simultaneously non-zero) are non-zero play a role in determining w and therefore enter Eq. (13). From Karush-Kuhn-Tucker (KKT) conditions, [15,16] it also follows that only for training data points for which the prediction error (*i.e.*, $\left|f\left(\vec{\mathbf{x}}_i\right)-y_i\right|$) is greater than ϵ, the Lagrange multipliers may be nonzero. Therefore, we have a sparse expansion of w in terms of $\vec{\mathbf{x}}_i$ (i.e. not using all $\vec{\mathbf{x}}_i$ to describe w). These non-vanishing coefficients are called Support Vectors.

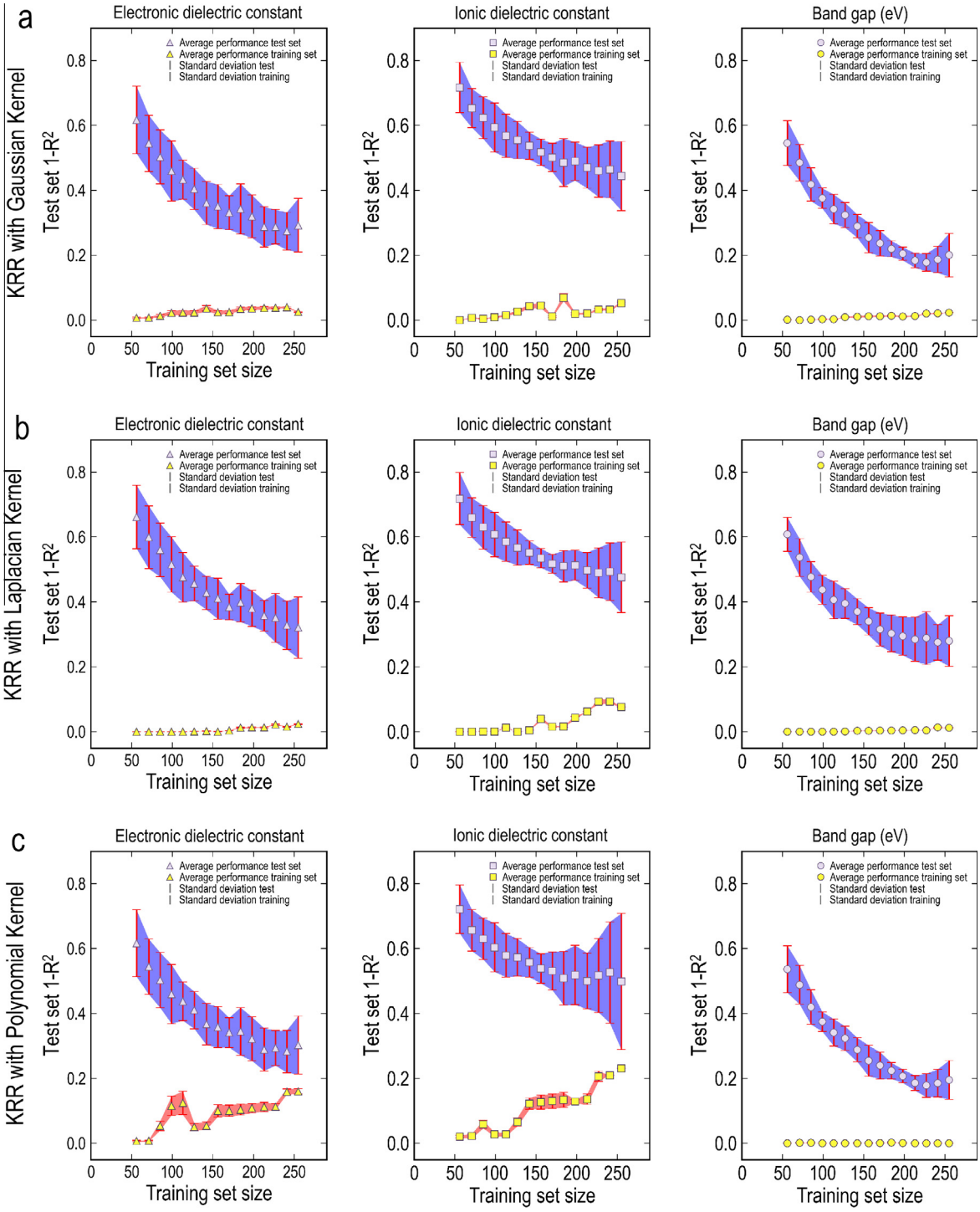

Fig. 5. Learning curves for KRR ML models with different kernels.

Thus far we have only considered a linear SVR problem. However, moving to a non-linear case from here is relatively straightforward and can be done by defining a kernel function $\Phi(x)$ that takes a point x in the feature space and transforms it non-linearly in the kernel space. Furthermore, since the SVR algorithm's dual optimization in Eq. (12) only depends on the dot products between patterns x_i, for the non-linear case, it should suffice to know the analogous dot product $\mathcal{K}(x, x')$ in the kernel space given by $\langle \Phi(x), \Phi(x') \rangle$. This allows us to restate the non-linear SVR optimization problem as:

$$\begin{aligned} \text{maximize} \quad & -\frac{1}{2}\sum_{ij=1}^{n}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)\mathcal{K}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) \\ & -\epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \quad & \sum_{ij=1}^{n}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]. \end{aligned} \tag{14}$$

Note that in the nonlinear setting, the optimization problem corresponds to finding the flattest function in feature space, not in input

space. Furthermore, to be an admissible kernel, $\mathcal{K}(x,x')$ is required to satisfy Mercer's condition. [17] Finally, following an analogous expression to the Eq. (13), predictions on new systems for the non-linear case can be made as:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)\mathcal{K}(\vec{\mathbf{x}}_i, x) + b. \quad (15)$$

We take the bias term b to be zero, which leaves the Kernel parameter and the tradeoff constant C as the parameters that need to be optimized. In our case, after analyzing initial test performance on several kernels (such as linear, polynomial and Gaussian; all of which are admissible SVR kernels), we decided to go forward with the Gaussian kernel. Our results for this kernel are presented in Figs. 6 and 7. Given that the parameters being optimized are σ and C, we measure model prediction errors for each combination of the two parameters and make plots showing the errors in Fig. 6, similar to Figs. 1–3. It can be seen that the optimal σ values are always around 1, whereas C takes different optimal values from 4 to 2^5. These respective pairs of optimal parameters were taken for the final SVR prediction models, whose performances are shown in Fig. 7.

It can be seen from the parity plots in Fig. 7 that the regression performances are slightly worse than with KRR using a Gaussian kernel (shown in Fig. 4a). The training and test set prediction errors for the three properties have been listed in Table 2. Whereas the training performances are worse for the electronic and ionic dielectric constants (than with Gaussian KRR), there is a clear problem of overfitting in the data for the band gap, which is again owing to the tradeoff constant C being higher (similar to the explanation for KRR with a polynomial kernel, in Section 2.1). Conventional SVR thus appears to not improve upon Gaussian kernel based ridge regression, and we attempt to rectify this in the following section with a technique known as 'AdaBoost'.

4. AdaBoost

Boosting refers to the general problem of coming up with an accurate prediction algorithm by optimally combining a number of weak learners. Belonging to this family, 'AdaBoost', short for 'Adaptive Boosting', is a Godel Prize winning machine learning technique that has commonly been applied in conjunction with regular regression algorithms (such as SVR, as we considered here) for improving their performances. [18,19] Boosting involves focusing on the particular points that have not been predicted well with SVR, that is, the difficult data points. If certain parameters could be modified so as to improve predictions on those points without affecting the predictions on all other points, we would have a better model than with regular SVR.

The AdaBoost algorithm is conceptually very simple. It is an iterative process where during each iteration, a new regressor is trained on the training set, with weights that are associated with each data point in the training set. These weights are modified at each iteration according to how successfully that data point has been predicted in the past iterations. The data points in the training set with larger prediction errors (*i.e.*, those that are difficult to predict) are assigned larger weights. In practice, for a regressor such as SVR the boosting procedure involves training the model a number of times by changing the parameters σ and C as explained above, such that we will have different models with different accuracies of prediction on the poorly predicted points. There may be some models where predictions are better than the others, and these models deserve special attention. The overall prediction model is reported as a weighted median of all these models (as discussed below), with higher weights given to the specific models where predictions on the difficult data points show low errors. This means that the result of boosting is an optimal prediction model that is a weighted combination of all the different models.

More specifically, the algorithm can be out lined as follows:

Input: an n–sample $S = \left\{\left(\vec{\mathbf{x}}_1, y_1\right), \left(\vec{\mathbf{x}}_2, y_2\right), \ldots, \left(\vec{\mathbf{x}}_N, y_N\right)\right\}$ training data, a learning algorithm $\Im$ and an integer T.

Initialize: For each training pattern we assign a weight $w_i = 1/N$, for $i \in [1, N]$

Do while the average loss $\bar{L}$ defined below is less than $\frac{1}{2}$ or $t < T$, maximum number of iterations:

- Set probability that training sample i is in the training set is $p_i^{(t)} = w_i^{(t)} \Big/ \sum w_i^{(t)}$ where the summation is over all N members of the training set.
- Pick N_1 samples (with replacement) to form the training set according to the probabilities $p_i^{(t)}$.
- Construct a regression machine from this training set with N_1 samples. Call the learning algorithm $\Im$ to get the hypothesis $h_t : x \to y$
- Calculate an average loss: $\bar{L}^{(t)} = \sum_{i=1}^{N_1} p_i^{(t)} L_i^{(t)}$, where $L_i^{(t)} = \frac{|h_t(\vec{\mathbf{x}}_i) - y_i|}{max|h_t(\vec{\mathbf{x}}_i) - y_i|}$ represents linear loss. In principle, other types of loss functions such as square loss or exponential loss can also be used here.
- Compute $\beta = \frac{\bar{L}_i^{(t)}}{1 - \bar{L}_i^{(t)}}$. β is a measure of confidence in the predictor. The lower the β the higher the confidence in the prediction.
- Update the weights: $w_i^{(t+1)} = w_i^{(t)} \beta^{1 - L_i^{(t)}}$. Note that smaller the loss, more is the weight reduction in the subsequent iteration, thereby making the probability small enough so that this pattern will be picked as a member of the training set for the next iteration.

Output: For a particular input x_i, each of the T machines makes a prediction h_t , with $t = 1, \ldots, T$. The final cumulative prediction h_f is made by using the T predictors as follows: $h_f = inf\left\{y \in Y : \sum_{t:h_t \leqslant y} log(1/\beta_t) \geqslant \sum \frac{1}{2} log(1/\beta_t)\right\}$, which is essentially the weighted median of the predictions from the T machines.

Based on the above algorithm as applied to our data, we obtain the cumulative predictions for every point, and Fig. 8 shows parity plots similar to Fig. 4. The respective training and test set errors obtained here are again listed in Table 2. It can be seen that while the training performances (as compared with regular SVR) have definitely improved with Boosting for the three properties, the test performance is only slightly better for the electronic dielectric constant and worse for the ionic dielectric constant and the band gap. This means that while boosting can possibly improve upon regular SVR, there are some points that are quite poorly predicted with SVR, especially for the ionic dielectric constant. Further, the test errors with SVR + Adaboost are still higher than the errors with KRR using a Gaussian kernel.

The performances with SVR and AdaBoost can be explained as a consequence of the nature of the data we have. Whereas typical materials science data mining problems would include large amounts of data, [3,20,21] our dataset of 284 polymers and their properties constitutes a 'small dataset'. The regression performances with both regular SVR and AdaBoost could be improved for a larger, more diverse set of polymers, where the fraction of poorly predicted points could perhaps be minimized. As such, KRR with Gaussian kernel is the algorithm that performs better on average than these techniques, as captured in Table 2, thus bringing a measure of redemption to the practices followed by us and others using materials science data in the recent past.

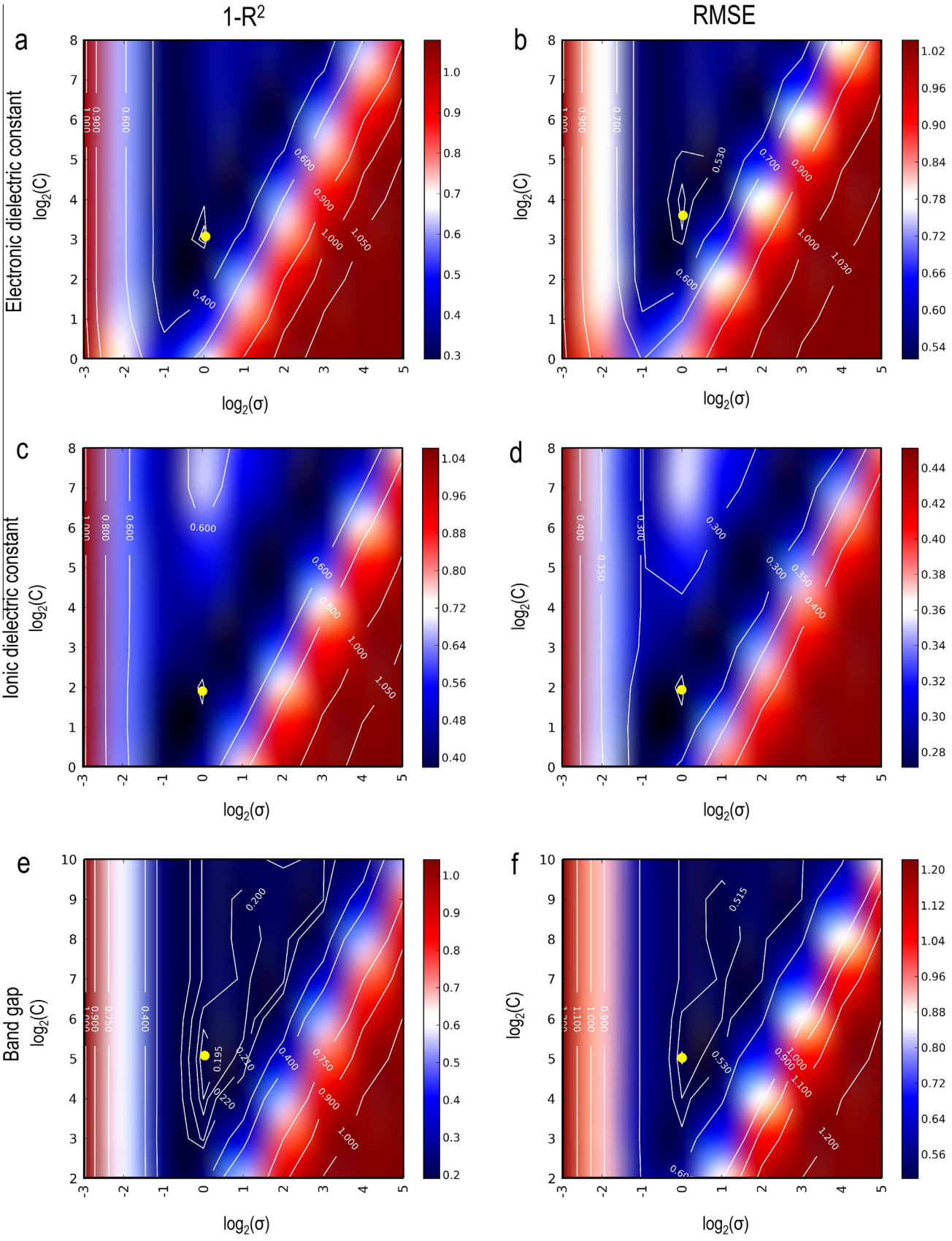

Fig. 6. Optimal parameter selection for the SVR ML models with Gaussian kernels.

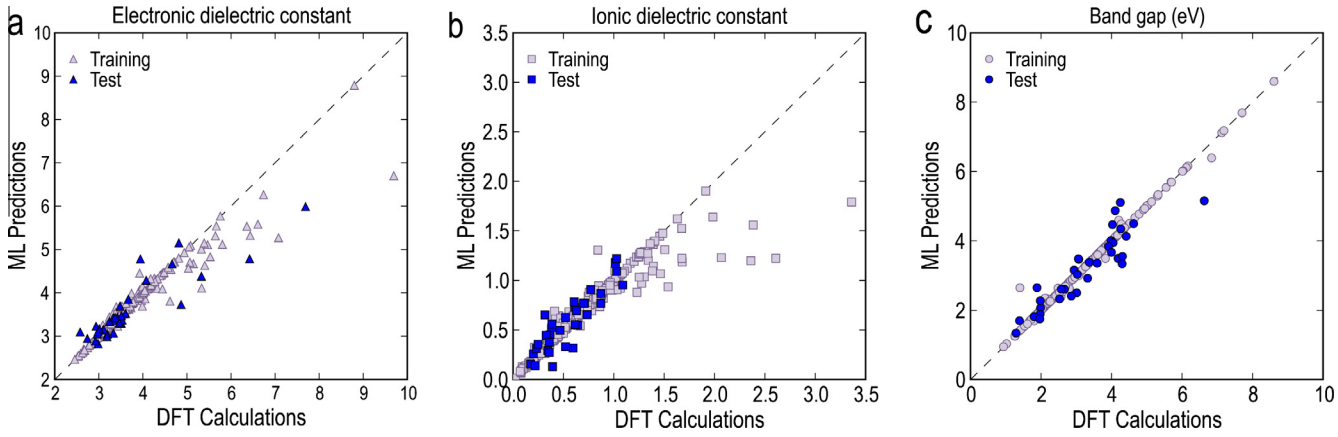

Fig. 7. Learning and prediction performances of the SVR ML models with a Gaussian kernel.

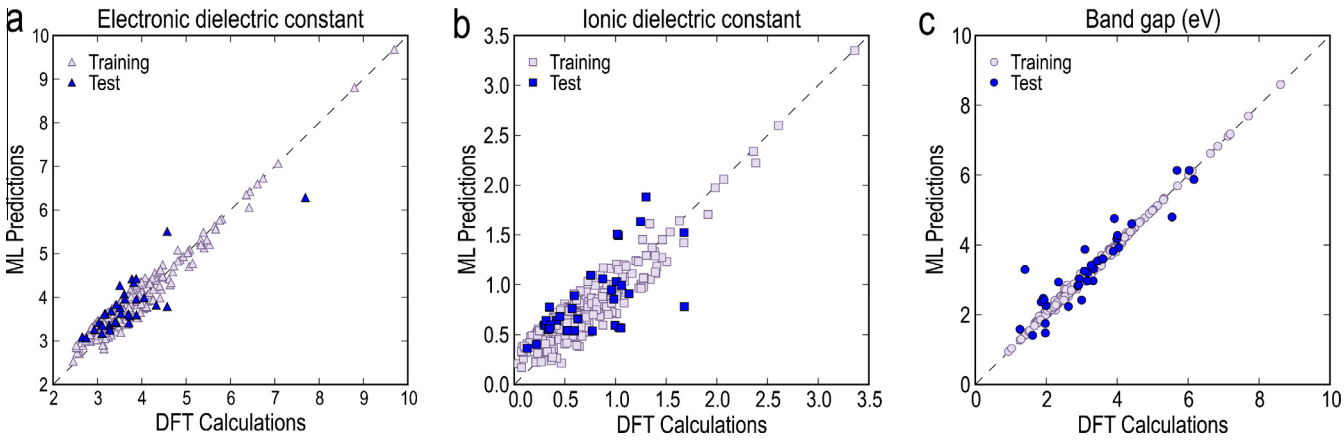

Fig. 8. Learning and prediction performances of the SVR ML models with ada boosting.

5. Conclusions

In conclusion, we applied different kinds of machine learning treatments on a dataset of organic polymers and gained some insight on the appropriate choices for learning parameters. Given our objective to develop accurate, robust prediction models by mapping polymer fingerprints (the input) to the properties (the output) using regression, we explored a number of different kinds of regression strategies. Regression performed using different learning algorithms, different distance kernel definitions and different training set sizes revealed that Kernel Ridge Regression with a Gaussian kernel and a sufficient training set size resulted in the best prediction performances. While KRR with a Laplacian kernel performed almost as well, the polynomial kernel appeared to be unsuitable. Another major algorithm, Support Vector Regression, was also used and it was seen that SVR, even when used with AdaBoost, did not improve upon the best KRR performance.

The prediction accuracies are limited by the size and nature of the computational data, as well as by the quality of the fingerprints used. Since the polymer fingerprint only takes the population of different combinations of chemical building blocks into account, factors such as the conformation and the planarity of polymer chains—which could have varying effects on the properties—are ignored here. While KRR with a Gaussian kernel appears to be the best regression algorithm to use on the given data, performances can further be improved with larger datasets and an improved fingerprint that contains more information than currently used.

Acknowledgements

This paper is based upon work supported by a Multidisciplinary University Research Initiative (MURI) grant (N00014-10-1-0944) from the Office of Naval Research. Computational support was provided by the Extreme Science and Engineering Discovery Environment (XSEDE) and the National Energy Research Scientific Computing Center (NERSC). G.P. acknowledges the support of the U.S. Department of Energy through the LANL/LDRD Program through a Director's postdoctoral fellowship. AMK would further like to thank LANL for providing computational resources.

References

[1] A. Mannodi-Kanakkithodi, G. Pilania, T.D. Huan, T. Lookman, R. Ramprasad, Machine learning strategy for accelerated design of polymer dielectrics, Sci. Rep. 6 (20952). http://dx.doi.org/10.1038/srep20952.
[2] G. Pilania, C.C. Wang, X. Jiang, S. Rajasekaran, R. Ramprasad, Accelerating materials property predictions using machine learning, Sci. Rep. 3 (2810). http://dx.doi.org/10.1038/srep02810.
[3] G. Pilania, A. Mannodi-Kanakkithodi, B.P. Uberuaga, R. Ramprasad, J.E. Gubernatis, T. Lookman, Machine learning bandgaps of double perovskites, Sci. Rep. 6 (19375). http://dx.doi.org/10.1038/srep19375.
[4] T.D. Huan, A. Mannodi-Kanakkithodi, R. Ramprasad, Accelerated materials property predictions and design using motif-based fingerprints, Phys. Rev. B 92 (2015) 014106, http://dx.doi.org/10.1103/PhysRevB.92.014106.
[5] T. Mueller, A.G. Kusne, R. Ramprasad, Machine learning in materials science: recent progress and emerging applications, in: Reviews in Computational Chemistry, John Wiley & Sons, Inc., 2016.
[6] K.T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K.R. Müller, E.K.U. Gross, How to represent crystal structures for machine learning: towards fast prediction of

electronic properties, Phys. Rev. B 89 (2014) 205118, http://dx.doi.org/10.1103/PhysRevB.89.205118.
[7] F. Faber, A. Lindmaa, O.A. von Lilienfeld, R. Armiento, Crystal structure representations for machine learning models of formation energies, Int. J. Quant. Chem. 115 (16) (2015) 1094–1101, http://dx.doi.org/10.1002/qua.24917.
[8] T.D. Huan, A. Mannodi-Kanakkithodi, C. Kim, V. Sharma, G. Pilania, R. Ramprasad, A polymer dataset for accelerated property prediction and design, Sci. Data 3 (160012). http://dx.doi.org/10.1038/sdata.2016.12.
[9] A. Mannodi-Kanakkithodi, G.M. Treich, T.D. Huan, R. Ma, M. Tefferi, Y. Cao, G.A. Sotzing, R. Ramprasad, Rational co-design of polymer dielectrics for energy storage, Adv. Mater. 28 (30) (2016) 6277–6291, http://dx.doi.org/10.1002/adma.201600377.
[10] V. Sharma, C. Wang, R.G. Lorenzini, R. Ma, Q. Zhu, D.W. Sinkovits, G. Pilania, A. R. Oganov, S. Kumar, G.A. Sotzing, S.A. Boggs, R. Ramprasad, Rational design of all organic polymer dielectrics, Nat. Commun. 5 (4845). http://dx.doi.org/10.1038/ncomms5845.
[11] K. Vu, J.C. Snyder, L. Li, M. Rupp, B.F. Chen, T. Khelif, K.-R. Mller, K. Burke, Understanding kernel ridge regression: common behaviors from simple functions to density functionals, Int. J. Quant. Chem. 115 (16) (2015) 1115–1128, http://dx.doi.org/10.1002/qua.24939.
[12] B. Schölkopf, The kernel trick for distances, in: Advances in Neural Information Processing Systems 13, Max-Planck-Gesellschaft, MIT Press, Cambridge, MA, USA, 2001, pp. 301–307.
[13] V. Vapnik, The nature of statistical learning theory, in: The Nature of Statistical Learning Theory, Springer-Verlag, New York, 2000.
[14] V. Vapnik, Statistical learning theory, in: Statistical Learning Theory, John Wiley and Sons, New York, 1998.
[15] W. Karush, Minima of Functions of Several Variables with Inequalities as Side Constraints, Master's thesis, Dept. of Mathematics, Univ. of Chicago, 1939.
[16] H.W. Kuhn, A.W. Tucker, Nonlinear programming, in: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, Calif., 1951, pp. 481–492.
[17] J. Mercer, Functions of positive and negative type, and their connection with the theory of integral equations, Philos. Trans. Roy. Soc. Lond. A: Math. Phys. Eng. Sci. 209 (441–458) (1909) 415–446, http://dx.doi.org/10.1098/rsta.1909.0016, arXiv: <http://rsta.royalsocietypublishing.org/content/209/441-458/415.full.pdf>.
[18] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: In Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufman, 1996, pp. 148–156.
[19] R. Rojas, Adaboost and the Super Bowl of Classifiers a Tutorial Introduction to Adaptive Boosting.
[20] V. Botu, R. Ramprasad, Adaptive machine learning framework to accelerate ab initio molecular dynamics, Int. J. Quant. Chem. 115 (16) (2015) 1074–1083, http://dx.doi.org/10.1002/qua.24836.
[21] F. Faber, A. Lindmaa, O.A. Von Lilienfeld, R. Armiento, Machine learning energies of 2 M elpasolite (ABC2D6) crystals.